Rec'd PCT/PTO 02 AUG 2005

WO 2004/073234 PCT/US2004/003299

10/544172

KEY EXPANSION FOR QKD

Claim of Priority

This patent application claims priority from U.S. Provisional Patent Application No. 60/445,805, filed on February 7, 2003.

Technical Field

The present invention relates to quantum cryptography, and in particular relates to key expansion methods applied to keys established between quantum key distribution (QKD) stations for the purpose of forming one-time pads for sending encrypted information between the QKD stations.

Background Art

Quantum key distribution involves establishing a key between a sender ("Alice") and a receiver ("Bob") by using weak (e.g., 0.1 photon on average) optical signals transmitted over a "quantum channel." The security of the key distribution is based on the quantum mechanical principal that any measurement of a quantum system in unknown state will modify its state. As a consequence, an eavesdropper ("Eve") that attempts to intercept or otherwise measure the quantum signal will introduce errors into the transmitted signals, thereby revealing her presence.

The general principles of quantum cryptography were first set forth by Bennett and Brassard in their article "Quantum Cryptography: Public key distribution and coin tossing," Proceedings of the International Conference on Computers, Systems and Signal Processing, Bangalore, India, 1984, pp. 175-179 (IEEE, New York, 1984). A specific QKD system is described in U.S. Patent No. 5,307,410 to Bennet (the '410 patent).

The Bennett-Brassard article and the '410 patent each describe a so-called "one-way" QKD system wherein Alice randomly encodes the polarization of single photons, and Bob randomly measures the polarization of the photons. The one-way system described in the '410 patent is based on two optical fiber Mach-Zehnder interferometers. Respective parts of the interferometer are accessible by Alice and Bob so that each can control the phase of the interferometer. The signals (pulses) sent from Alice to Bob are time-multiplexed and follow different paths.

U.S. Patent No. 6,438,234 to Gisin (the '234 patent), which patent is incorporated herein by reference, discloses a so-called "two-way" QKD system that is autocompensated for polarization and thermal variations. The two-way system is based on a folded interferometer wherein the optical pulses traverse the same path through
the interferometer, but with a time-delay.

The general operation of a QKD system is described in the book by Bouwmeester, Ekert and Zeilinger (Eds.) entitled "The physics of quantum information," Springer-Verlag (2001), section 2.3. In the operation of the two-way phase-encoding system of the '234 patent, Bob generates a single optical pulse and forms therefrom two coherent pulses P1 and P2 that travel to Alice with a time delay between the pulses. Alice attenuates the pulses to make them weak and then randomly phase modulates one of the pulses (say P1). Alice also reflects the pulses with a Faraday mirror so that the polarization of each pulse is rotated by 90° before returning to Bob. The pulses return to Bob and in doing so traverse the same round-trip path through the interferometer but in different order. Bob then randomly phase-modulates the yet-unmodulated pulse P2, and recombines the now-modulated pulses P1 and P2. The combined pulses interfere, and the result of the interference is detected in one of two detectors, depending on the respective phases imparted to pulses P1 and P2 by Alice and Bob, respectively. The detected pulses constitute Bob's measured qubits.

After a sufficiently large number of qubits are exchanged between Bob and Alice, Bob and Alice publicly compare the basis used to encode each photon, and also discard photons that did not arrive at Bob or Alice. Alice and Bob keep only those qubits corresponding to the same phase-encoding basis. This forms the sifted key. Alice and Bob then choose at random some of the qubits in the sifted key to test for errors that reveal the presence of an eavesdropper. These test qubits are then discarded. If there are no errors, the remaining qubits form the shared key. At this point, an operation called "privacy amplification" is typically performed. This operation involves deducing the number of bits (ζ) by which the (error-corrected) key k_N of n bits needs to be shortened so that any information an eavesdropper has about the final key k_F is lower than a specified value. Privacy amplification further includes forming a binary matrix K of dimension $(n - \zeta) \times n$, publicly sharing this matrix, and then performing $k_F = K \cdot k_N \pmod 2$ to arrive at the final key k_F.

Finally, an authentication step is typically performed wherein a previously shared authentication key or code is used to ensure that the Alice is really Alice and Bob is really Bob. This authentication step can be performed at any point in the key exchange process.

Using the above-described processes, the final key k_F has a given length. The length of the key k_F dictates the amount of information that can be encrypted. The secure key rate from a QKD system is usually too low for commercially available data transmission lines if one-time pad encryption is being used. Also, the achievable data bandwidth is very low and is limited by the key generation rate, which is around 1-10 kbps with present technology. The technology of the present invention, set forth below, presents a method that allows QKD to encrypt broadband streams of data (up to 10 Gbps or more), thus providing a method for expanding the key without having to send additional photons over the system.

Brief Description of the Drawings

FIG. 1 is a flow diagram that provides an overview of the QKD processes of the present invention;

FIG. 2 is a flow diagram illustrating key sifting as run on Alice's computer;

FIG. 3 is a flow diagram that continues from the flow diagram of FIG. 2, and which illustrates key shuffling using a modified Cascade protocol on Alice's computer;

FIG. 4 is a flow diagram that continues from the flow diagram of FIG. 3, and that illustrates an example embodiment of generating an error-free key on Alice's computer, based on performing privacy amplification of the shuffled key;

FIG. 5 is a flow diagram that illustrates an example embodiment of performing key sifting on Bob's computer;

FIG. 6 is a flow diagram that continues from the flow diagram of FIG. 5, and that illustrates an example embodiment of key shuffling using a modified Cascade protocol on Bob's computer;

FIG. 7 is a flow diagram that continues from the flow diagram of FIG. 6, and that illustrates an example embodiment of generating an error-free key on Bob's computer by performing privacy amplification of the shuffled key;

FIG. 8 is a flow diagram illustrating an example embodiment of performing key expansion on a privacy amplified key;

FIG. 9a is a diagram of an exemplary key schedule for "Pad Expansion Flag" = 0, for the case when AES in CTR mode is used; and

FIG. 9b is a diagram of an exemplary key schedule for "Pad Expansion Flag" = 1, for the case when AES in CTR mode is used.

Detailed Description of the Invention

The present invention has industrial utility in the fields of quantum cryptography, quantum key distribution, and data encryption. In particular, the present invention has industrial utility in combining quantum cryptography and classical cryptography. The present invention provides protection from not only eavesdroppers that utilize passive tapping of transmitted information from a carrier such as, but not limited to, an optical photon, but from any type of intrusion attack including involved active types of attacks wherein an eavesdropper probes the Alice and Bob nodes (stations) using a probe signal sent through an optical fiber used to transmit data.

As described in greater detail below, the present invention includes a method for generating a cryptographically secure key between two stations. An example method includes exchanging single photon signals between two QKD stations to establish a plurality of matching raw keys at each station. The method also includes processing the raw keys using error correction and privacy amplification, to establish matching privacy amplified keys at each station. The method also includes buffering the privacy amplified keys at each station to form matching key schedules. The method further includes forming at least one expanded key from a key selected from the key schedules, wherein the expanded key serve as a one-time pad to encrypt information to be exchanged between the two stations.

The discussion below assumes that the eavesdropper ("Eve") is unlimited in her technological resources. The system is optimized to provide maximal key generation rate in unconditional security regime. As an example, the system described herein can be implemented on a Pentium III, 500 MHz machine or better machines (assuming that quantum layer clock does not exceed 10 MHz), and also on a digital signal processor (DSP). A communication channel of approximately 10 Mbit/s may be utilized (depending on quantum layer clock rate and optical channel length).

Also, in the discussion below, the term "key" is interchangeable with "pad"

when the pad is the same length as the key. However, generally speaking, a key is used to form a pad, say by generating a string of bits to be used for data encoding. In this case, the pad has a different length (i.e., number of bits) than they key and is not the same as the key *per se*. However, in the discussion below, an "expanded key" is used as a pad for one-time pad encryption. Also, the term "expanded pad" is used below and is the same as the "expanded key." The term "expanded pad" is used to denote a pad that is expanded over a pad that would be formed from an *unexpanded* key.

In an example embodiment, the method of present invention includes a number of main algorithmic parts, including sifting, error correction, privacy amplification and key expansion (or no key expansion). The error correction protocol is optimized to reveal as few bits on the public channel as possible. In addition, the error correction protocol utilizes encryption (preferably one-time pad) of bits sent over the public discussion (PD) layer.

The privacy amplification method utilized herein implements a multiplication by a random binary matrix. The matrix multiplication is preceded by cryptographically strong sifted key shuffling. The key expansion step uses a stream cipher.

QKD process overview

FIG. 1 is a flow diagram that provides a general overview of the QKD process according to the present invention. With reference to FIG. 1, to start a quantum key distribution (QKD) process, the two parties involved (Alice and Bob) share initial secret information that is required for authentication of a sifting procedure. In the sifting stage, the two parties involved in QKD exchange basis and single-photon detector 'click' information.

In order to 'fight quantum memory' the sifting process is started with a delay of $\tilde{>}1$ s after bits are detected. Bob (the party ultimately receiving photons) sends Alice (the party sending photons) information about click time slots and basis information for positions of the clicks. Alice responds with a stream of bits encoding the 'correct' bases for the 'clicks'. Both parties disregard the bits with 'wrong' bases.

The sifting procedure is authenticated. For this purpose, Alice and Bob form a string of bits that they are sending/receiving in pre-agreed form and calculate message authentication code (MAC) values of the string. If the latter values

coincide, Alice and Bob proceed to the error correction protocol. If the MAC values to not coincide, they issue an intrusion or "attack alert."

Authentication may be performed by means of any strong authentication procedure. Preferable procedures are the unconditionally secure message authentication codes (for example, UMAC).

Other possible authentication procedures can be based on hash function SHA, RIPEMD (or other) based HMAC. The latter case is not unconditionally secure. However, the parties can decide to encrypt/partially encrypt the MAC by using a one-time pad with the absolutely secure key they possess. This removes any possibility of cracking the authentication. It is preferable to use some kind of encryption for the whole sifting procedure, for example AES or TDES.

With continuing reference to FIG. 1, the sifted key is buffered until it reaches a number of bits high enough (*e.g.*, 10^5 bits or more) to run the error correction protocol. Cryptographically strong shuffling is then performed on the sifted key stored in the buffer. Both parties (i.e., Alice and Bob) use the same seed for the shuffling procedure, which they take from their absolutely secure key buffer. In this way, an eavesdropper has no information on a shuffling result. The seed is refreshed at pre-agreed time moments. The shuffling step is utilized to randomize the positions of errors that are needed for realization of an efficient error correction protocol, to erase (preferably to a very high degree) eavesdropper's information on specific bit positions in the sifted key, and to prepare the sifted key for the privacy amplification procedure.

The shuffling step is a security enhancement to the standard BB84 protocols and in an example embodiment may be skipped if all the data transmitted over public channel is encrypted.

The error correction utilized by the system is characterized, among other ways, by the party correcting the errors deciding on a number of passes of a Cascade protocol based on a hash value provided by another party. After each pass through the Cascade protocol, the party calculates the hash value on its buffer and compares it to the received value. When the hash values coincide, the Cascade is stopped. If the values do not coincide, then another pass of Cascade is performed. In this way, fewer bits have to be sent over the public channel, and the probability of failure of Cascade to correct some errors is effectively zero. In general, any hash function with good mixing properties may be used (*e.g.*, SHA, MD5).

Another modification of the Cascade process includes encrypting parity bits sent over the public discussion layer. The hash value mentioned above should preferably be encrypted as well. Therefore, an eavesdropper listening to the communication over the public discussion layer (channel) does not receive information on bit parities. It should be noted that information on specific bit positions is, to a high degree, erased in the shuffling step.

The outcome of the error correction procedure is buffered for privacy amplification. This buffering may be required to adjust different sizes of blocks in the error correction and privacy amplification steps. Privacy amplification is performed as multiplication of the error-corrected string by a fixed random binary matrix M. The shuffling step enables maintaining the matrix as fixed during QKD.

Quantum key generation at Alice

FIG. 2 [Alice-1], FIG. 3 [Alice-2], and FIG. 4 [Alice-3] are flow diagrams of a quantum key generation process according to the present invention as performed by Alice's computer, located at the Alice node. Further "Alice's apparatus", "Alice's computer" and "Bob's apparatus" and "Bob's computer" are referred to as "Alice" and "Bob," respectively.

FIG. 2 [Alice-1] is a flow diagram illustrating a sifting stage as run on Alice's computer. In 8-1, Alice receives quantum layer key bits and bases from the RNG within Alice's apparatus. In 8-2, Alice receives click positions of single photon detectors at Bob's apparatus (CPB), bases for clicks (BCB), and message authentication code (MACB1) from Bob. In 8-3, Alice calculates her message authentication code (MACA1) and compares it to Bob's message authentication code (MACB1). In 8-4, MACB1 is compared to MACB2. If the values of MACB1 and MACA1 coincide ("yes"), Alice concludes that there was no attempt to tamper with data. If the values of MACB1 and MACA1 do not coincide ("no"), the program generates an attack alert at 8-5 and sends a warning to Bob at 8-6.

In the case that values of MACB1 and MACA1 coincide, in 8-7 Alice forms a correct basis positions string (CBPA) by checking bases for coincidence and ~~calculates~~ a message authentication code for CBPA (MACA2). In 8-8, CBPA and MACA2 are then sent to Bob. Thereafter, in 8-9, Alice picks bits that correspond to correct bases and discards the rest of the bits.

FIG. 3 [Alice-2] is a flow diagram that continues from the flow diagram of FIG.

2, and contains buffering for sifting stage and shuffling a modified Cascade, as an example, a protocol part of the algorithm as run on Alice's computer. Following 8-9 of FIG. 2, buffering is completed in 9-1. Then in 9-2, Alice performs cryptographically strong shuffling utilizing the pre-agreed part of the key. Then in 9-3, the shuffled string hash (ha) is calculated and in 9-4 ha is sent to Bob.

For error correction, Alice uses the modified Cascade protocol with one-time pad parity bit encryption at 9-5. Within each run of 9-5, an Nth pass of modified Cascade algorithm is communicated to Bob. After each pass of the modified Cascade protocol of 9-5, Alice checks the status of coincided hash values sent from Bob. If at 9-6 a new hash (h) and the old hash (ha) at Bob's side do not coincide ("no"), a higher pass of the modified Cascade is run. The process is stopped when coinciding hashes ("yes") are achieved at 9-6. In 9-7, results of error correction may also be buffered.

The parties might decide to exchange some authenticated information about their generated keys to ensure that there was no spoofing during communication over the public channel.

FIG. 4 [Alice-3] is a flow diagram that continues from the flow diagram of FIG. 3 and that illustrates the generation of error-free key stage as run on Alice's computer. In 10-1, privacy amplification is performed on the corrected bit string by multiplication of the corrected bit stream by a fixed random binary matrix M. This procedure generates the secure key. In 10-2, the secure key is buffered synchronously to Bob's station and in 10-3 the secure key is outputted.

Buffering is a good measure against denial of service (DOS) attack. In case of DOS attack, the stations issue an alarm to the control center. For example, the stations can send a simple network management protocol (SNMP) trap. The stations can then start using buffered keys instead of QKD-generated keys until the DOS is mitigated. The size of the buffer should be big enough to provide the time for technical personnel to address the DOS problem. It should be noted that standard implementation of quantum cryptography is not resistant to DOS attacks.

Quantum key generation process at Bob

FIG. 5 [Bob-1], FIG. 6 [Bob-2], and FIG. 7 [Bob-3] are flow diagrams diagram for a quantum key generation process as performed by Bob's computer. FIG. 5 [Bob-1] is a flow diagram that contains a sifting stage as run on Bob's computer. In

11-1, Bob receives quantum layer bases for clicks from RNG within Bob's apparatus (BCB) and click positions (CPB) from the single photon detector(s) through, for example, a 32 bit input/output port. In 11-2, a string is formed for CPB and for BCB. In 11-3, for these strings Bob calculates message authentication code (MACB1). In 11-4, Bob sends Alice BPB, BCB and MACB1. In turn, in 11-5 Bob receives from Alice CBPA and MACA2. In 11-6, for the data received, message authentication code (MACB2) is once more calculated, and in 11-7 MACB2 and MACA2 are compared. If MACB2 and MACA2 do not coincide ("no"), an attempt of tampering with data occurred during data transfer. In this case, the in 11-8 program generates an attack alert and in 11-9 sends a warning to Bob and Alice. If MACB2 and MACA2 do coincide ("yes") at 11-7, no attack happened during data transfer. In this case, in 11-10 the program picks bits that correspond to correct bases and discards the rest of the bits.

FIG. 6 [Bob-2] is a flow diagram that continues from the flow diagram of FIG. 5 and that contains buffering for a sifting stage and shuffling, and a modified Cascade protocol part of the algorithm as run on Bob's computer.

In 12-1, the bits selected in 11-10 are buffered. After buffering is completed, then in 12-2 cryptographically strong shuffling utilizing the pre-agreed part of the key is performed. In 12-3, Bob receives hash from Alice and saves it (ha) in 12-4. For error correction, in 12-5, Bob uses the modified Cascade protocol with one-time pad parity bit encryption. Within each run of 12-5, an Nth pass of the modified Cascade algorithm communication with Bob takes place. Such data exchange assures parity bit encryption. After each pass of the modified Cascade protocol of 12-5, a new hash (h) is calculated in 12-6, and compared to the previous hash (ha) in 12-7. If h and ha do not coincide ("no"), a higher pass of the modified Cascade is run in 12-5. The process is stopped when coinciding hashes are achieved ("yes") at 12-6. In this case Bob, in 12-8 sends Alice a flag, which informs Alice that hash h and hash ha coincided. In 12-9, results of error correction are also buffered.

FIG. 7 [Bob-3] is a flow diagram that continues from the flow diagram of FIG. 6 and contains a generation of error-free key stage as run on Bob's computer. In 13-1 (which follows buffering step 12-9), privacy amplification is performed on the corrected bit string by multiplication of the corrected bit stream by a fixed random binary matrix M. This procedure generates the secure key. The secure key from 13-1 is then buffered in 13-2 and a key is generated in 13-3.

Again, in case of a DOS attack, the stations issue an alarm to the control center and follow the above-described procedure to ensure continued operation.

Key expansion

It should be noted that, in an example embodiment of the present invention, an arbitrarily long key for one-time pad encryption is produced, which increases the encrypted data bandwidth. Strictly speaking, one-time pad-like encryption with the expanded key is no longer one-time pad encryption. However, all the operations a user has to perform to encrypt the data are identical to the operations needed to implement one-time pad encryption, so the term 'one-time pad' is used herein.

As mentioned above, the secure key rate from a QKD system is usually too low for commercially available data transmission lines if one-time pad encryption is being used. At the same time, in many cases using a one-time pad is an excess security measure. Accordingly, the following describes a QKD device that comprises a key expansion protocol in addition to the standard protocols of QKD. This lets a user select between two regimes of QKD operation: one-time pad and one-time expanded pad. In principle, a user can expand a key only at the user's site. However such a solution is expensive and not necessarily safe, thus reducing the security level of the QKD system. The present technique allows users to choose between an expanded key regime and a one-time pad regime. However, the key expansion itself is performed within the QKD apparatus.

Current implementations of QKD suffer from a low key rate that today is on the order of 1 kbit/sec. The key rate is defined by the clock rate of the QKD device, imperfections of light sources, finite efficiency of detectors, finite finesse of optical elements, and losses in the optical fiber connecting the QKD stations. Though technological advances will likely enable an increase in the rate of secure bits, this rate will be lower than the bandwidth of commercially available communication lines. This prevents using QKD systems with one-time pad encryption on such communication lines.

On the other hand, one-time pad is a natural encryption choice when a QKD is used. A one-time pad ~~ensures ultimate~~ security, thus making one-time pad encryption and QKD combination unbreakable. The drawback is that using a one-time pad limits the data transmission bandwidth to that of the key rate.

In many cases, one-time pad encryption is excessive and other classical

symmetric key encryption techniques can be employed with a QKD system. These techniques would require additional hardware and/or software. The present invention concerns a QKD system that does not provide specialized encryptors. Such systems can be used in one-time pad encryption mode. The following describes a feature of a QKD system ("one-time pad/ expanded pad switch") that enables the system to be used in either one-time pad or expanded pad regimes. The one-time pad embodiment provides the ultimate security delivered by quantum key distribution, while the expanded pad regimes solves the problem of finite data bandwidth in cases when one-time pad is not required.

A one-time pad/expanded pad switch can readily be added to a QKD system. In many cases, it can be realized by software means on existing QKD hardware. Preferably, both one-time pad and expanded pad regimes use the same key distribution interface.

The QKD system can be seen as a perfect random number generator producing the same random bits at two remote locations. At the switch position 'one-time pad,' the generated bits are sent via interface to the user and for one-time pad encryption of a particular message.

At a switch position 'expanded pad' the random bits serve as a seed for a cryptographically strong pseudo random number generator that can be implemented on QKD hardware to produce a pseudo random bit stream that is long enough to encrypt a message and which, again, is sent via the same interface, to the user and for one-time pad encryption of the message.

The communicating parties switch the regimes synchronously. For synchronization, they can use any communication network in their possession. For example, the party – Alice or Bob - that requests key expansion, sets "Pad Expansion Flag" = 1 for a pre-agreed bit, and exchanges this bit with another party within the accepted communication protocol between the two QKD stations. The synchronization protocol also can be implemented in a QKD device and can use the communication line of QKD. The party transmitting the encrypted message informs the receiving party of the encryption regime and, in case of expanded pad regime, the key expansion ratio.

Example of key (pad) expansion

Key expansion can be performed in a QKD device after the privacy

amplification protocol, or it can be combined with privacy amplification step. FIG. 8 [Bob-31] is a flow diagram for the embodiment where key expansion is performed after privacy an amplification protocol, represented by 14-1. The privacy amplified key is buffered in 14-2. In 14-3, the QKD system generates random bit blocks r=r1, r2, ..., rN. The cryptographically strong pseudorandom number generator P stretches the bits blocks pi=P(ri), producing a pseudorandom bit stream p=p1,p2,...,pN. For the pseudorandom number generator, one may use a type of stream cipher, for example AES in CTR mode. The pseudorandom bit stream is used for one-time pad encryption in expanded pad mode. This operation is run on both QKD stations, i.e. at Bob and Alice. In 14-4, parties check the value of the exchanged "Pad Expansion Flag" bit. It should be noted here that the term "Pad Expansion" is used to denote expanding the key to form the (expanded) pad.

If "Pad Expansion Flag"=0, then the output is just a key as generated during the QKD process. Having buffered keys is mind, the key schedule with AES in Counter (CTR) mode has a structure "ID_1 Key_1; ID_2 Key_2; ID_N Key_N" (see FIG. 9a), where ID_I is a number assigned to each key Key_I.

In 14-5, the expanded pad is outputted. If "Pad Expansion Flag"=1, than the output is expanded key. Than the key schedule with AES in CTR mode has a structure "ID_1 Key_1 Pad_1; ID_2 Key_2 Pad_2;...:ID_N Key_N Pad N" (see FIG. 9b), where ID_I is a number assigned to each pair of Key_I and Pad_I. Pad_N is a bit stream generated by means of AES-256 (Key_N) using AES in CTR mode. There is no correlation of ID_N with either Key_N or Pad_N. The key expansion ratio depends on the user's data bandwidth and the key generation rate. Key data is never reused. Although a key ID may be reused, its associated key data will always be different.

Thus, in the present invention one-time pad encryption need not be used to encrypt data whose bandwidth is higher than the key generation rate. In other words, if the data bandwidth B is higher than the key generation rate K, then we use K in combination with some classical cryptographic methods. For example, K is split into n 256 bit blocks- k1, k2,..., kn (n=K/256) (in that way n is a number of 256 bit keys we can generate per second) and keys kj are used with some classical encryption method to encrypt data for time 1/n seconds

Similar implementation can be achieved for other cipher modes as well. For example, in CBC mode one would need to send the Key_ID only and there is no

need to send the offset. CBC mode provides a higher data integrity check. If, for example, the packets are being encrypted at OSI level 2, then the data forgery after decryption will usually be noticed by OSI level 3 protocols. In that way CBC mode provides much better data integrity check than CTR mode. To implement CBC mode encryption, a user might need specialized hardware.

It is worth noting that AES in CTR (as well as many other block stream ciphers) has a known flaw in that it is subject to data forgery or spoofing. Because of this, it is preferred that it be used with a data authenticity mechanism to ensure that the data has not been altered with out authorization.

The same implementation can be used with other stream ciphers. It is worth noting that the stream ciphers, which provide message authentication, can prevent data forgery and spoofing.

Pad expansion can be performed by cryptographically strong pseudorandom generators (like block stream ciphers). The privacy amplification step utilizes is a QKD protocol that produces a perfectly secure bit stream from a partially secure bit stream. This is usually achieved by applying some universal hash function on the partially secure bits (*e.g.*, Bennet, Brassard- Generalized PA). The hash function compresses the bit stream in a way so that eavesdropper has exponentially little information on the compressed bits. To some extent, pad expansion 'does the opposite.' Specifically, pad expansion stretches the bit stream so that more key bits are received at the output.

The expanded pad can be generated by modifying the privacy amplification procedure by adjusting the compression ratio. The latter reduces computational complexity and, thus, computational resources of the hardware. As an example, one possible embodiment involves the communicating parties applying a cryptographically strong shuffling procedure on partially secure bits without compression. The seed for the shuffling procedure can be exchanged between the parties over any existing communication network that the parties possess. The parties can agree on a one-time pad encryption for the shuffling seed. Starting at this point, the parties can seed a cryptographically strong pseudo random number generator with the shuffled bits.

In accordance with an example embodiment of the present invention, the receiver of the single-bit transmission is aware of when the transmitter is going to transmit the encrypted data to the receiver. It should be noted that in one example

the transmitter is transmitting the encrypted data to the receiver, while in another example the receiver may instead be a transmitter, while the transmitter acts as a receiver.

User data encryption

In an example embodiment, the present invention is used perform data encryption in the following as manner outlined below. At the encryption station (say, Bob), the station reads the user data packet and determines the packet data size in bytes. The encryption station then checks to see if the sum of the current offset and the packet data size is less than the Pad size in bytes. If not, the station jumps to the next key ID and resets the offset to zero.

The encryption station then XORs the packet data with an appropriate pad, starting from current offset value. It adds the Key ID and offset information in unencrypted format to the packet. The latter can be done by encapsulating the packet into another packet, by adding an additional header, or by using some already existing field of the data packet. The encryption station then increases the current offset by the size of the encrypted data packet and transmits the encrypted packet to the receiver.

At the decrypting station (say, Alice), the packets are read and the Key ID and offset are extracted. The received data is then XORed with the appropriate pad starting from the received offset value. The receiver station then restores all packet headers (if necessary), and passes the data to the user.